Description of SINGULAR
A Computer Algebra System for Singularity Theory, Algebraic Geometry and Commutative Algebra

Gert-Martin Greuel

Contents

1 WHY A NEW SYSTEM? 2

2 WHAT IS SPECIAL? 2

3 HOW TO GET SINGULAR? 3

4 WHAT IS SINGULAR ABOUT? 3

4.1 Baserings and ground fields . . . 3
4.2 Main algorithms . . . 4
4.3 Monomial orderings . . . 5
4.4 Programming language . . . 5
4.5 Libraries . . . 5
4.6 Links . . . 5

5 APPLICATIONS AND EXAMPLES 6

5.1 How to enter and exit . . . 6
5.2 Defining a ring . . . 6
5.3 Solving equations . . . 7
5.4 Elimination of variables . . . 7
5.5 Use of parameters . . . 8
5.6 Free resolutions . . . 9
5.7 Deformation of singularities . . . 10

6 WHAT WILL COME? 11

SINGULAR has been created and its development is being directed and coordinated by G.-M. Greuel, G. Pfister and H. Schönemann with contributions by O. Bachmann, H. Grassmann, B. Martin, W. Neumann, W. Pohl, T. Siebert and R. Stobbe.

In the following we describe some of the main features of SINGULAR and give some examples of how to use it.

1 WHY A NEW SYSTEM?

The development of SINGULAR started in 1986 by G. Pfister and H. Schönemann in order to be able to compute standard bases in local rings or, more precisely, in the localization of the polynomial ring at the origin. The main algorithm for doing this was Mora's modification of Buchberger's algorithm to compute Gröbner bases in polynomial rings. To explain the difference between these two algorithms let us think about a system of polynomial equations in many variables having only finitely many solutions. Using Buchberger's algorithm it is possible to compute the total number of all solutions counted with multiplicities, while Mora's algorithm allows us to compute the multiplicity of a single, specified solution. The need for this local version of Buchberger's algorithm arose from research problems in pure mathematics, trying to find a counterexample for complete intersections of a theorem of K. Saito about the exactness of the Poincaré complex of hypersurfaces. Indeed, using this implementation, a counterexample was found.

None of the existing systems offered the possibility for such local computations. This was the starting point of (a forerunner of) SINGULAR. It was mainly able to compute certain special invariants of singularities. In September 1991 the SINGULAR project became a joint venture of the Humboldt University of Berlin and the University of Kaiserslautern. Since 1994 the development of SINGULAR has been continued exclusively at the University of Kaiserslautern.

The main stimulans for enlarging the system, by including new algorithms, comes from research problems arising from alglebraic geometry, commutative algebra and singularity theory. Since about one year there exists a close cooperation between the SINGULAR group in Kaiserslautern and a group from the Centre for Microelectronics at the University of Kaiserslautern, in which SINGULAR is going to be applied in a system which is designed for sizing analog electric circuits. This collaboration has some influence on the further development and implementation of special strategies for certain algorithms in SINGULAR.

2 WHAT IS SPECIAL?

Many systems (such as the commercial systems Axiom, Maple, Mathematica, Macsyma, Reduce, ..., or non–commercial systems such as CoCoA, Macaulay, GB, POSSO, ...) offer the possibility to compute Gröbner bases in polynomial rings.

The commercial systems are usually not very fast and are limited in the choice of orderings, and offer little more than just the possibillity to compute a Gröbner basis. The non–commercial systems usually specialize in the purposes for which they were created and, for example, have only a limited class of ground fields, do not offer the possibility to compute with parameters or with modules over a polynomial ring and, most seriously, have no method to compute efficiently in local rings. SINGULAR provides, among other things, the following features:

- Computations in very general rings, including polynomial rings, localizations hereof at a prime ideal and tensor products of such rings. This includes, in particular, Buchberger's and Mora's algorithm as special cases.
- Many ground fields for the above rings, such as the rational numbers, finite fields Z/p, (p a prime ≤ 32003), finite fields with $q = p^n$ elements, transcendental and algebraic extensions, floating point real numbers with single precision.

- Usual ideal theoretic operations, such as intersection, ideal quotient, elimination and saturation and more advanced algorithms based on free resolutions of finitely generated modules. Several combinatorial algorithms for computing dimensions, multiplicities, Hilbert series ....
- A programming language, which is C-like and which is quite comfortable and has the usual if-else, for, while, break ... constructs.
- Library of procedures, written in the SINGULAR language, which are useful for many applications to mathematical problems.
- Links to communicate with other systems or with itself. This communication is based on MP (by S. Gray, N. Kaijler, P. Wang) and MPP (by O. Bachmann and H. Schönemann), a protocoll designed for both a very general and a very efficient exchange of polynomial data.
- Last, but not least, SINGULAR is designed for speed. Although it has a very general standard basis algorithm, it is quite fast for computations of Gröbner bases for, say, homogeneous ideals in polynomial rings over fields of small characteristic and it seems to belong to the fastest systems for computations over the rationals.

3 HOW TO GET SINGULAR?

SINGULAR is avaiblable via anonymous ftp from

helios.mathematik.uni-kl.de.

(log in as ftp with your email address as password.) The directory /pub/Math/Singular/bin contains the binaries for different machines, the manual as tex file, the online help, the libraries and two useful files "some.tips" and "singular–algorithms.tex".

Singular runs on several platforms: SUN Sparc, HP9000/300 and HP9000/700, Silcon Graphics, Dec alpha,IBM RS/6000, Next (m68k based), Linux, Macintosh, MSDOS, ... Please read the README file!

Copyright by G.-M. Greuel, G. Pfister, H. Schönemann, All Rights Reserved. SINGULAR may be copied and distributed freely for any non–commercial purpose. If you copy SINGULAR for somebody else, you may ask for a refund of your expenses. You are not allowed to ask for more than this. In any case you must include a copy of this copyright notice.

Parts of SINGULAR have their own copyright like the GNU MP and the MP library. See the Reference Manual for details.

If you obtain SINGULAR please send a short notice containing your full name and address via e–mail to singular@mathematik.uni-kl.de. This allows us to keep track of the users of SINGULAR and you will profit from new versions, new libraries, etc. Please report bugs to the same adress.

If you publish a mathematical or any other result that was partly obtained using SINGULAR, please cite SINGULAR. Also we would appreciate it if you could inform us about such a paper.

4 WHAT IS SINGULAR ABOUT?

4.1 Baserings and ground fields

Almost all computations in SINGULAR require a basering which may be a

- polynomial ring,

- series ring, i.e. a localization of a polynomial ring,
- factor ring by an ideal of one of the above,
- tensor products of one of the above,
- exterior algebra,
- tensor product of one of the above rings with an exterior algebra,
- Weyl algebra, D-modules.

(The last three algebras require a special compiled version with DRING resp. SRING.)

The ground field of these algebras may be

- the rational numbers Q (char 0),
- a finite field Z/p, p a prime ≤ 32003 (char p),
- a finite field with $q = p^n$ elements ($p^n \leq 2^{15}$),
- transcendental extension $K(A, B, C, \dots)$, $K = Q$ or Z/p,
- algebraic extension $K[Z]/\mathrm{MinPol}$, $K = Q$ or Z/p,
- floating point real numbers with single precision (real).

4.2 Main algorithms

The basic algorithm in SINGULAR is a general standard basis algorithm for any monomial ordering which is compatible with the natural semigroup structure of the exponents. This includes wellorderings (Buchberger algorithm) and tangent cone orderings (Mora algorithm) as special cases.

Advanced techniques such as Traverso's Hilbert–driven Gröbner basis algorithm or the weighted–ecart–method and the high–corner–method, developed by the SINGULAR group.

Algorithms for the usual ideal theoretic operations, such as intersection, ideal quotient, elimination and saturation.

Different syzygy algorithms (sres, res, mres) for computation of free resolutions of modules over the above rings.

Combinatorial algorithms for computing dimensions, Hilbert series, multiplicities, etc.

Factorization of multivariate polynomials over the rationals and algebraic extensions.

Algorithms for primary decomposition of ideals and modules.

Algorithms for computing invariants of singularities or affine or projective varieties such as Milnor, Tjurina and discriminate numbers, singular locus, T1 and T2, Ext groups etc.

Algorithms for computing a semiuniversal deformation of an isolated singularity or a module up to a given order.

4.3 Monomial orderings

The most important monomial orderings in SINGULAR are:

- global or p-orderings: lp, dp, wp, Dp, Wp refer to a polynomial ring,
- local or s-orderings: ls, ds, ws, Ds, Ws refer to a series ring,
- matrix orderings: allow to create any ordering by a matrix,
- product (or block) orderings of the above.

The great variety of different orderings and their efficient implementation make SINGULAR a powerful tool for many different tasks in Algebraic Geometry, Algebra and Singularity theory.

4.4 Programming language

SINGULAR has a comfortable programming language similar to C with for, if ... else, while, break, continue, etc. This language my be used by the user to write procedures which enlarge the system by some functions for his own need. At the moment these procedures are interpreted but it is planned to precompile them in a later version.

4.5 Libraries

SINGULAR contains several libraries, with procedures written in the SINGULAR language. The user may add his own procedures/libraries. At the moment the following libraries are distributed with SINGULAR:

inout.lib	procedures for manipulating in– and output
general.lib	procedures of general type
deform.lib	procedures for computing miniversal deformation
elim.lib	procedures for elimination, saturation and blowing up
factor.lib	procedures for calling the REDUCE factorizer (UNIX only)
homolog.lib	procedures for homological algebra
matrix.lib	procedures for matrix operations
poly.lib	procedures for manipulating polys and ideals
random.lib	procedures for random matrix and poly operations
ring.lib	procedures for manipulating rings and maps
sing.lib	procedures for singularities

In one of the next versions we shall also add the libraries

solve.lib	procedures for solving polynomial equations
invar.lib	procedures for computing invariants of algebraic groups
primedec.lib	procedures for primary decomposition

More libraries will be added and the existing libraries will be enlarged and updated continually.

4.6 Links

Links describe the communication channels of SINGULAR, i.e. something SINGULAR can read from and/or write to. The following links exist (under developement, not all are implemented in all versions):

Via Ascii links any data can be written into files for storage or communication with other programmes. Data exchange via ascii links is not the fastest way, but the most general.

DBM links provide access to data stored in a data base (using the UNIX dbm or gbdm library).

MP file links give the possibility to store data in binary format. Read and write access is very fast compared to ascii links.

MP TCP links give the possibility to exchange data in binary format between two processes which may run on the same or different computers via tcp sockets. Read and write access is very fast compared to ascii links.

Based on these links, SINGULAR is able to communicate with other systems or with itself. At the moment a communication is implemented between SINGULAR-SINGULAR, SINGULAR-FACTORY (a C++ library for multivariate polynomials) SINGULAR-Mathematica and, on a lower level, SINGULAR-Maple.

5 APPLICATIONS AND EXAMPLES

5.1 How to enter and exit

To start SINGULAR, enter Singular at the system prompt. You will get a header with the version name, the compilation date and a short remark how to get help on SINGULAR. Enter help; to get the online manual at the SINGULAR prompts. All input is case–sensitive.

To leave SINGULAR, type one of the following: quit; exit; exit all; $

5.2 Defining a ring

Almost all computations in SINGULAR need a ring. Every number, polynomial, vector, ideal, module and every matrix in SINGULAR must be defined over a base ring which is a polynomial ring over a field or a localization hereof (depending on the monomial ordering, see above) or a quotient ring (factor ring) hereof modulo a standard basis. The ring (resp. qring for quotient ring) command informs SINGULAR about

- the coefficient field,
- the names of indeterminants and
- the monomial ordering.

Let us consider a few examples:

ring $R1 = 32003, (t, x, y, z), dp$; defines a poynomial ring with name $R1$ over the field with 32003 elements, 4 variables t, x, y, z and monomial ordering dp (for degree reverse lexicographical ordering, p refers to polynomial ring).

ring $R2 = (0, A), x(1..10), ds$; defines a ring with name $R2$ over the field $Q(A)$ where Q denotes the rationals, 10 variables $x(1), \ldots, x(10)$ and monomial ordering ds (for degree reverse lexicographical ordering, s refers to series ring) which is the localization of $Q(A)[x(1), \ldots, x(10)]$ at the maximal ideal. Here A is an independent parameter. But if we set, for example, minpoly $= A^2 + A + 3$;, we get an algebraic extension of Q defined by minpoly.

ring $R3 = (181, a, b, c, d), (x, y, z, x(1..5)), (dp(3), ds)$; defines a ring over the field $Z/181(a, b, c, d)$ with four independent parameters a, b, c, d, with eight variables $x, y, z, x(1), \ldots, x(5)$ and product ordering dp for the first 3 variables and ds for the remaining variables. Note that the product ordering realizes the tensor product of the polynomial ring in x, y, z and the localization of the polynomial ring in $x(1), \ldots, x(5)$.

ring $R4 = (2^5, s), (x, y, z), wp(1, 2, 3)$; defines a ring over the Galois field with 32 elements and s the generator of its multiplicative group, with variables x, y, z having weights 1,2,3 and (weighted) degree reverse lexicographical ordering.

5.3 Solving equations

SINGULAR is designed for treating systems of nonlinear polynomial equations. A simple task is to solve linear equations symbolically. For eample, solve the following system of linear equations where a, b, c, d are free parameters (a, b, c, d may be of course replaced by numbers):

$$\begin{array}{rrrr} 3x+ & y+ & z- & u = a \\ 3x+ & 8y+ & 6z- & 7u = b \\ 14x+ & 10y+ & 6z- & 7u = c \\ 7x+ & 4y+ & 3z- & 3u = c \end{array}$$

In SINGULAR this can be achieved by defining an appropriate ring, an ideal and compute a standard basis:

$$\begin{array}{rcl} ring\ r & = & (0, a, b, c, d), (x, y, z, u), dp; \\ ideal\ i & = & 3x + y + z - u - a, \\ & & 13x + 8y + 6z - 7u - b, \\ & & 14x + 10y + 6z - 7u - c, \\ & & 7x + 4y + 3z - 3u - d; \end{array}$$

```
option(redSB);    // set the option to compute a reduce standard basis
ideal j = std(i); // compute a standard basis, call it j
j;                // display the result
```

The solutions can be read off from the output (set the right–hand side equal 0):
$j[1] = u + (6/5 * a + 4/5 * b + 1/5 * c - 12/5 * d)$
$j[2] = z + (16/5 * a - 1/5 * b + 6/5 * c - 17/5 * d)$
$j[3] = y + (3/5 * a + 2/5 * b - 2/5 * c - 1/5 * d)$
$j[4] = x + (-6/5 * a + 1/5 * b - 1/5 * c + 2/5 * d)$

For systems of nonlinear equations the same method may be applied but with a lexicographical ordering lp. The result of a standard basis computation will be a set of polynomials in a kind of triangular form. If the equations have only finitely many solutions, this reduces the problem to solving a polynomial equation in one variable.

5.4 Elimination of variables

Elimination is one of the problems which is encountered in almost any kind of application of Gröbner basis techniques. The geometric interpretation of elimination of certain variables from an ideal J is the following: the set of zeroes of the ideal obtained after elimination is the (closure of the) image of the projection of the set of zeroes of J to the subspace without the eliminated variables.

The following example stems from a problem in microelectronics, the sizing problem of a (very simple) amplifier, where internal currents, voltages, resistors etc. have to be eliminated:

ring $r = 0, (a, b, c, d, e, f, g, h, i, j, k, l, m, x, y), (dp(11), lp)$;
ideal $J = 10000x - y + l, y + 13000a + 100b - l, 1000c + d - m,$
197100090b-199091d-199091000e+219000m,
$-b + f, -a + e, -g + l, d - 5, y - h, y - i, h - j, i - k - m,$
$1600000f2 - 80000fj + 52000f - j, 1600000e2 - 80000ek + 52000e - k;$
eliminate($J, abcdefghijklm$);

The result is an implicit equation in x, y (input–voltage and output–voltage):
$\longrightarrow -[1] = -4239405858918400000000000000000000x4$
$\longrightarrow -2830363383855462400000000000000000x3y + 2380343414764656640000000000000000x3$
$\longrightarrow -525221597328171520000000000000000x2y2 + 2509736436685869262656000000000000x2y$
$\longrightarrow -157974362252865033616000000000000x2 - 19238163980006400000000000000xy3$

$\longrightarrow +128536020362679101436800000000xy2 - 1565995517758256108095520000 0xy$
$\longrightarrow +532361107154218377843600000 00x + 31502493517260480000000y3$
$\longrightarrow -188600655707336289307960y2 + 220408498981428298021221y$
$\longrightarrow -7248338358439822823 4000$

5.5 Use of parameters

The following innocent example produces in its standard basis extremely long coefficients in char 0 for the lexicographical ordering. But a very small deformation does not. We may realize the deformation symbolically by an extra deformation parameter. In both cases we compute the vectorspace dimension over the ground field, which shows that for a general value of the parameter the system of equations has less solutions than for the parameter $t = 0$. WARNING: in general the use of parameters is very time consuming.

option(prot); // shows a short protocol during std–computation
ring $R0 = 0, (x, y), lp$;
poly $f = x5 + y11 + xy9 + x3y9$;
ideal i = jacob(f); // ideal of partial derivatives of f
ideal $i1 = i, i[1] * i[2]$; // undeformed ideal
$i1$;

$\longrightarrow i1[1] = 5x4 + 3x2y9 + y9$
$\longrightarrow i1[2] = 9x3y8 + 9xy8 + 11y10$
$\longrightarrow i1[3] = 45x7y8 + 27x5y17 + 45x5y8 + 55x4y10 + 36x3y17 + 33x2y19 + 9xy17 + 11y19$
ideal j = std($i1$);
$\longrightarrow V19 - s20.s21(1)s22(2)23 - s27.s28.s29.s30.s31.s32.s33.s34.s35.s36.s37.s38.s39$
$\longrightarrow .s40.s.70-$
$\longrightarrow$ product criterion:1 chain criterion:30
j;
$\longrightarrow j[1] = 264627y39 + 26244y35 - 1323135y30 - 131220y26 + 1715175y21 + 164025y17 + 1830125y16$
$\longrightarrow j[2] = -121039477919718467198383218863933929137500650608 75xy8$
$\longrightarrow +286391521141683198701331939250003266767738632875y38$
$\longrightarrow +319544022069090269267646228775735657855443067259 1y37$
$\longrightarrow -574366214208226638497213812657388952828463 20y36$
$\longrightarrow -1657764214948799497573918210031067353932439400y35$
$\longrightarrow -213018481589308191195677223898986826970012055 00y34$
$\longrightarrow -182219415866306656558599197696156571964806980614 8y33$
$\longrightarrow +4701709279892816135156972313196394005220175y32$
$\longrightarrow +1351872269688192267600786976008506868242319 75y31$
$\longrightarrow +387306330592981081696151697602503805300114137 5y30$
$\longrightarrow -1325886675843874047990382005421144061861290080000y29$
$\longrightarrow -1597720195476063141946794589554240608952696688731 0y28$
$\longrightarrow +262701813363090926606333480026253304267126525y27$
$\longrightarrow +758608269089333526902713624894485954472795312 5y26$
$\longrightarrow +8678530741064946460228584335167214896539594562 5y25$
$\longrightarrow +55458081432735941021732523311518357002788639247 45y24$
$\longrightarrow -190755630134604373646791537790383948956383 25y23$
$\longrightarrow -548562322715501761058348996776922561074021125y22$
$\longrightarrow -157465452677648386073957464715681007809339831 25y21$
$\longrightarrow +14142791297211762229786542358173595055551911562 50y20$
$\longrightarrow +2071119006944589361521339965003571537816994342312 5y19$
$\longrightarrow -272942733337472665573418092977905322984009750y18$
$\longrightarrow -789065115845334505801847294677413365720955375 0y17$
$\longrightarrow -635548970384916867877296560610447246510898031 25y16$
$\longrightarrow +2209925172992390669973224476102826607435025596162 5y14$

$\longrightarrow -14793713967965590435357948972258591339027857296625y10$
$\longrightarrow j[3] = 5x4 + 3x2y9 + y9$

vdim(j); // the vectorspace dimension (k-dimension) of $R0/j$ over the ground field Q
$\longrightarrow$ 63

Let us now deform the above ideal by introducing a parameter t and compute over the ground field $Q(t)$.

$$\begin{array}{lcl} ring\ Rt & = & (0,t),(x,y),lp; \\ poly\ f & = & x5+y11+xy9+x3y9; \\ ideal\ i & = & \text{jacob}(f); \\ ideal\ j & = & i, i[1]*i[2]+t*x5y8; \text{// deformed ideal} \\ j; & & \end{array}$$

$\longrightarrow j[1] = (5)*x^4 + (3)*x^2*y^9 + y^9$
$\longrightarrow j[2] = (9)*x^3*y^8 + (9)*x*y^8 + (11)*y^{10}$
$\longrightarrow j[3] = (45)*x^7*y^8 + (27)*x^5*y^{17} + (1*t+45)*x^5*y^8 + (55)*x^4*y^{10}$
$\longrightarrow +(36)*x^3*y^{17} + (33)*x^2*y^{19} + (9)*x*y^{17} + (11)*y^{19}$
$j = std(j)$;
vdim(j); // the vectorspace dimension (k–dimension) of Rt/j over the ground field $Q(t)$
$\longrightarrow$ 40
kill $Rt, R0$;

5.6 Free resolutions

In SINGULAR a free resolution of a module is a list of modules. The first module in the list is the input module (eventually minimized). Of course some modules may be 0.

sres(i,r), mres(i,r), res(i,r) compute the first r modules of the resolution of i, resp. the full resolution if $r = 0$. NOTE: In SINGULAR the command ''betti" does not require a minimal resolution for the minimal betti numbers.

// Two transversal cusps in P^3:

ring $r3 = 0,(x,y,z,h),dp$; // global ordering dp
ideal $i = z2 - 1y3 + x3y, xz, -1xy2 + x4, x3z$;
i = homog(i,h); // homogenize i with homogenizing variable h
list resh = mres(i,0); // computes a minimal resolution
print(resh); // the full minimal resolution

$\longrightarrow$ [1]:
$\longrightarrow$ _[1] $= xz$
$\longrightarrow$ _[2] $= x3y - 1y3h + z2h2$
$\longrightarrow$ _[3] $= x4 - 1xy2h$
$\longrightarrow$ [2]:
$\longrightarrow$ _[1] $= zh2*gen(1) - 1x*gen(2) + y*gen(3)$
$\longrightarrow$ _[2] $= -1x3*gen(1) + y2h*gen(1) + z*gen(3)$
$\longrightarrow$ [3]:
$\longrightarrow$ _[1] $= 0$
intmat B=betti(resh); // create an intmat with graded betti numbers
print(B,"betti"); // this gives a nice output of betti numbers

$\longrightarrow$ 0 1 2
$\longrightarrow$
$\longrightarrow$ 0 : 1 0 0
$\longrightarrow$ 1 : 0 1 0
$\longrightarrow$ 2 : 0 0 0
$\longrightarrow$ 3 : 0 2 2
$\longrightarrow$
$\longrightarrow$ total : 1 3 2

The entry d at (i, j) is the minimal number of generators in degree $i + j$ of the j–th syzygy module of R^n/M (the 0th (resp. 1st) syzygy module of R^n/M is R^n (resp. M)). Hence, M has 1 generator in degree 2 and 2 generators in degree 4 while the 1st syzygy module of M (= module of relations of generators of M) has 2 generators in degree 5 (gen(1) has degree 2, gen(2), gen(3) degree 4).

5.7 Deformation of singularities

The libraries "sing.lib" resp. "deform.lib" contain procedures to compute the miniversal deformation of an isolated complete intersection singularity resp. arbitrary isolated singularity.

We give two examples, the first being a hypersurface, the second an arbitrary singularity. We compute for these the miniversal deformations:

LIB "deform.lib"; // be sure that SingularPath is set correctly so
// that SINGULAR can find the library deform.lib.

// 1. hypersurface case (from series $T[p, q, r]$):

ring $R = 32003, (x, y, z), ds$;
int $p, q, r = 3, 3, 4$;
poly $f = x^p + y^q + z^r + xyz$;
print(deform(f));

$\longrightarrow 1, x, xz, y, yz, z, z2, z3$
// the miniversal deformation of f is given by the projection from the
// miniversal total space to the miniversal base space:
// $\{(A, B, C, D, E, F, G, H, x, y, z)|x3 + y3 + xyz + z4 + A + Bx + Cxz + Dy + Eyz + Fz + Gz2 + Hz3 = 0\}$
// -$\longrightarrow$ $\{(A, B, C, D, E, F, G, H)\}$

// 2. general case (cone over rational normal curve of degree 4):

ring $R1 = 0, (x, y, z, u, v), ds$;
matrix $m[2][4] = x, y, z, u, y, z, u, v$;
ideal i =minor($m, 2$); // 2x2 minors of matrix m
miniversal(i,"Def_r","A("); // Def_r will be the name of the miniversal
// base space with parameters $A(1), \dots, A(4)$

$\longrightarrow$// dim T1 = 4
$\longrightarrow$// dim T2 = 3
$\longrightarrow$// computation finished in degree 2
$\longrightarrow$
$\longrightarrow$// — Equations of miniversal base space —
$\longrightarrow$ jet$J[1] = -1 * A(1) * A(4)$
$\longrightarrow$ jet$J[2] = A(3) * A(4)$
$\longrightarrow$ jet$J[3] = -1 * A(2) * A(4) - 1 * A(4)^2$
$\longrightarrow$
$\longrightarrow$// — Equations of miniversal total space —
$\longrightarrow$ jet$F[1] = -1 * y^2 + x * z + A(2) * x - 1 * A(3) * y$
$\longrightarrow$ jet$F[2] = -1 * y * z + x * u - 1 * A(1) * x - 1 * A(3) * z$

$\longrightarrow$ jet$F[3] = -1 * y * u + x * v - 1 * A(3) * u - 1 * A(4) * z$
$\longrightarrow$ jet$F[4] = -1 * z^2 + y * u - 1 * A(1) * y - 1 * A(2) * z$
$\longrightarrow$ jet$F[5] = -1 * z * u + y * v - 1 * A(2) * u - 1 * A(4) * u$
$\longrightarrow$ jet$F[6] = -1 * u^2 + z * v + A(1) * u - 1 * A(4) * v$
$\longrightarrow$
$\longrightarrow$// Result belongs to ring Def_r (total space of miniversal deformation).
$\longrightarrow$// Make Def_r the basering and list objects defined in Def_r by typing:
setring Def_r ; show(Def_r);
listvar(ideal);
killall(); // a proc from general.lib, killing all user–defined variables

6 WHAT WILL COME?

SINGULAR will be continually enlarged and updated. One of the next versions will contain:

- A factorizer of univariate and multivariate polynomials.
- A factorizing Gröbner algorithm (for solving systems of polynomial equations).
- Different algorithms for primary decomposition for ideals and modules.
- Graphical interface.

(A test version of SINGULAR with these three topics realized exists already.)

- A library realizing the Arnold classifier of singularities.
- FGLM–techniques for fast change of ordering.
- Trace algorithm for speeding up computations in characteristic 0.
- Links to other general purpose systems (Maple, Macsyma, …).
- Puiseux–expansion of plane (and space) curve singularities.
- Special functions allowing very fast computation in semigroup rings.
- Parallel standard basis computations (Alyson Reeves).
- Compilation of libraries